The background of the slide is a collage of images. At the top right, there is a computer mouse. Below it, a hand is holding a CD-ROM. In the center, there is a document with the word 'return' visible. The title 'Unidade 5' is in a teal box, and 'Armazenamento e Indexação' is in a white box.

Unidade 5

Armazenamento e Indexação

Engenharia de Computação / Engenharia de Produção

Banco de Dados

Prof. Maria das Graças da Silva Teixeira

Material base: Banco de Dados, 2009.2, prof. Otacílio José Pereira

Contexto na Disciplina

- 1 – Introdução
- 2 – Modelo Relacional
- 3 – SQL
- 4 – Projeto de Banco de Dados
- **5 – Armazenamento e Indexação**
- 6 – Processamento e Otimização de Consultas
- 7 – Gerenciamento de Transações
- 8 – Controle de Concorrência
- 9 – Recuperação de Falhas
- 10 – Segurança de Banco de Dados
- 11 – Tópicos Avançados

Reflexões Preliminares

- Quando realizamos uma consulta ou uma alteração, o SGBD é responsável por encontrar os dados e fazer a sua manipulação;
- Mas como os dados são armazenados internamente / fisicamente nos arquivos dos bancos de dados?

Roteiro

- **Conceitos básicos**
- **Índices Ordenados**
- **Arquivos indexados com árvore B e B+**
- **Definindo índices em SQL**

Conceitos Básicos

- **Índices** são estruturas de dados auxiliares que permitem agilizar a busca em tabelas nos bancos de dados;
- Questionamento:
 - Em um livro, como você recupera uma página de leitura sabendo-se o assunto/tópico que se deseja ler?

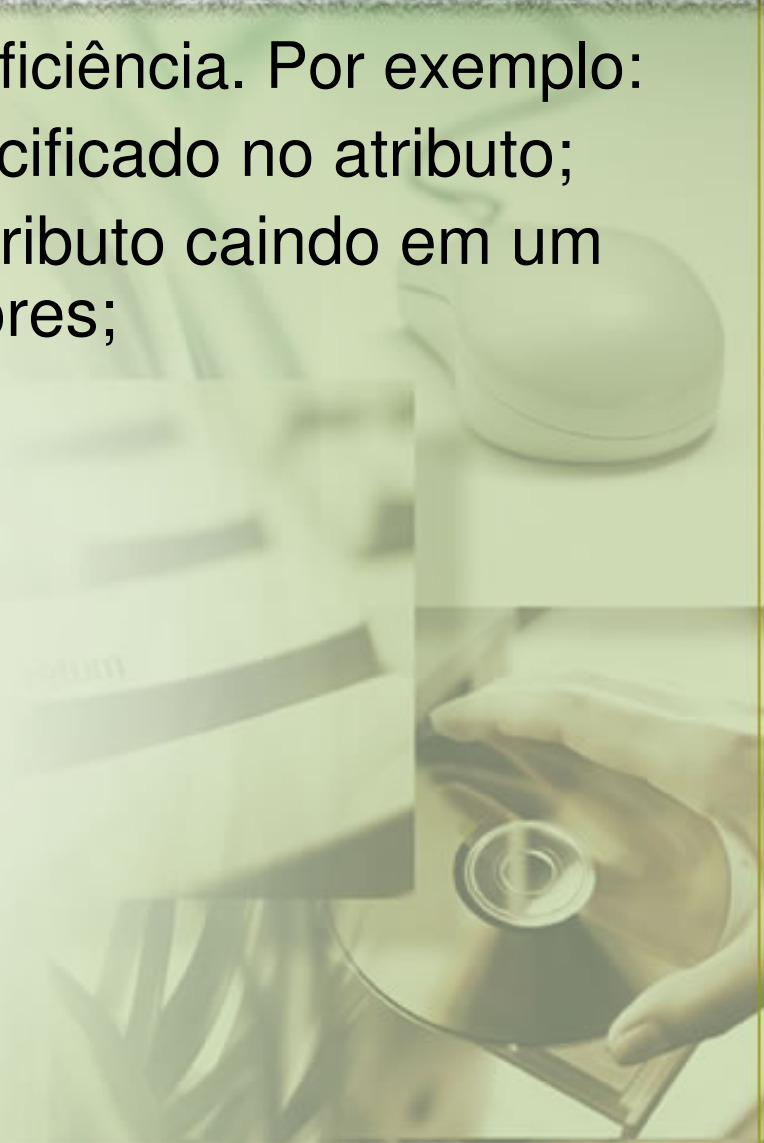
Conceitos Básicos

- Existem variados mecanismos de indexação usados para agilizar o acesso aos dados desejados. Por exemplo, um catálogo de autores em uma biblioteca;
- **Chave de Busca** - atributo ou conjunto de atributos para pesquisar registros em um arquivo;
- Um **Arquivo de Índice** consiste em registros (chamados **Entradas de Índice**) na forma:

| | |
|----------------|----------|
| Chave de busca | Ponteiro |
|----------------|----------|

- Arquivos de índice normalmente são muito menores do que o arquivo original – que contém os dados;
- Existem dois tipos básicos de índices:
 - **Índices Ordenados**: chaves de busca são armazenadas em ordem classificada;
 - **Índices de Hash**: chaves de busca são distribuídas uniformemente entre “baldes” usando uma “função de hash”.

Métricas de Avaliação de Índices

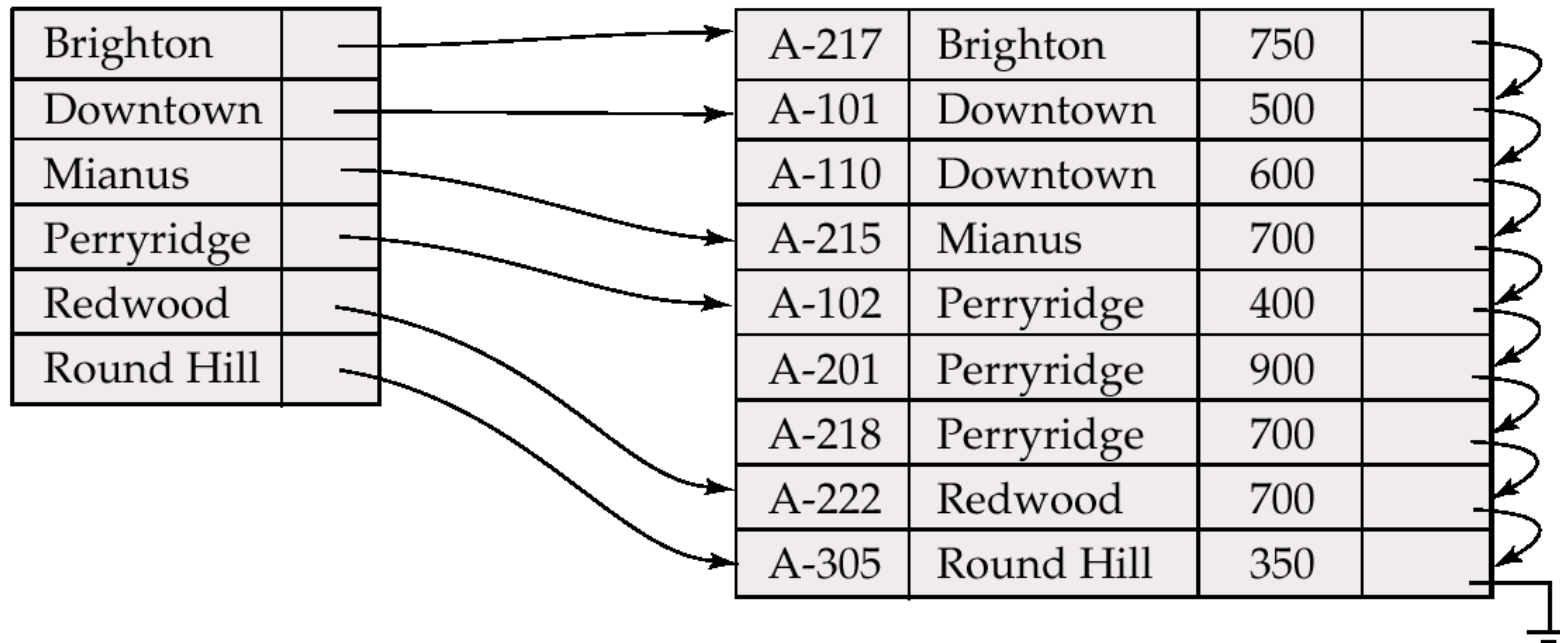
- Tipos de acesso admitidos com eficiência. Por exemplo:
 - Registros com um valor especificado no atributo;
 - Registros com um valor de atributo caindo em um intervalo especificado de valores;
 - Tempo de acesso;
 - Tempo de inserção;
 - Tempo de exclusão;
 - Sobrecarga de espaço.
- 

Índices Ordenados

- Em um índice ordenado, as entradas de índice são armazenadas classificadas pelo valor da chave de busca;
- **Índice Primário** - em um arquivo ordenado sequencialmente, é o índice cuja chave de busca especifica a ordem sequencial do arquivo. Também chamado **Índice de Agrupamento**;
 - A chave de busca de um índice primário normalmente, mas não necessariamente, é a chave primária;
- **Índice Secundário**: um índice cuja chave de busca especifica uma ordem diferente da ordem sequencial do arquivo. Também chamado **Índice Não de Agrupamento**;
- **Arquivo Sequencial Indexado**: arquivo sequencial ordenado com um índice primário.

Arquivos de Índice Denso

- **Índice Denso** — O registro de índice aparece para cada valor de chave de busca no arquivo.
- Por exemplo:



Arquivos de Índice Esparso

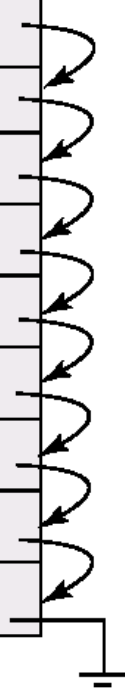
- **Índice Esparso:** contém registros de índice somente para alguns valores de chave de busca. Pode ser aplicado quando os registros são ordenados sequencialmente por chave de busca;
- Para localizar um registro com o valor de chave de busca K :
 - Encontramos o registro de índice com o maior valor de chave de busca que seja $< K$;
 - Pesquisamos o arquivo sequencialmente, começando no registro para o qual o registro de índice aponta;
- Requer menos espaço e causa menos sobrecarga de manutenção para inserções e exclusões do que os índices densos;
- Geralmente é mais lento do que o índice denso para localizar registros;
- Boa escolha: índice esparso com uma entrada de índice para cada bloco no arquivo, correspondente ao menor valor de chave de busca no bloco.

Arquivos de Índice Esperso

- Exemplo:

| | |
|----------|--|
| Brighton | |
| Mianus | |
| Redwood | |

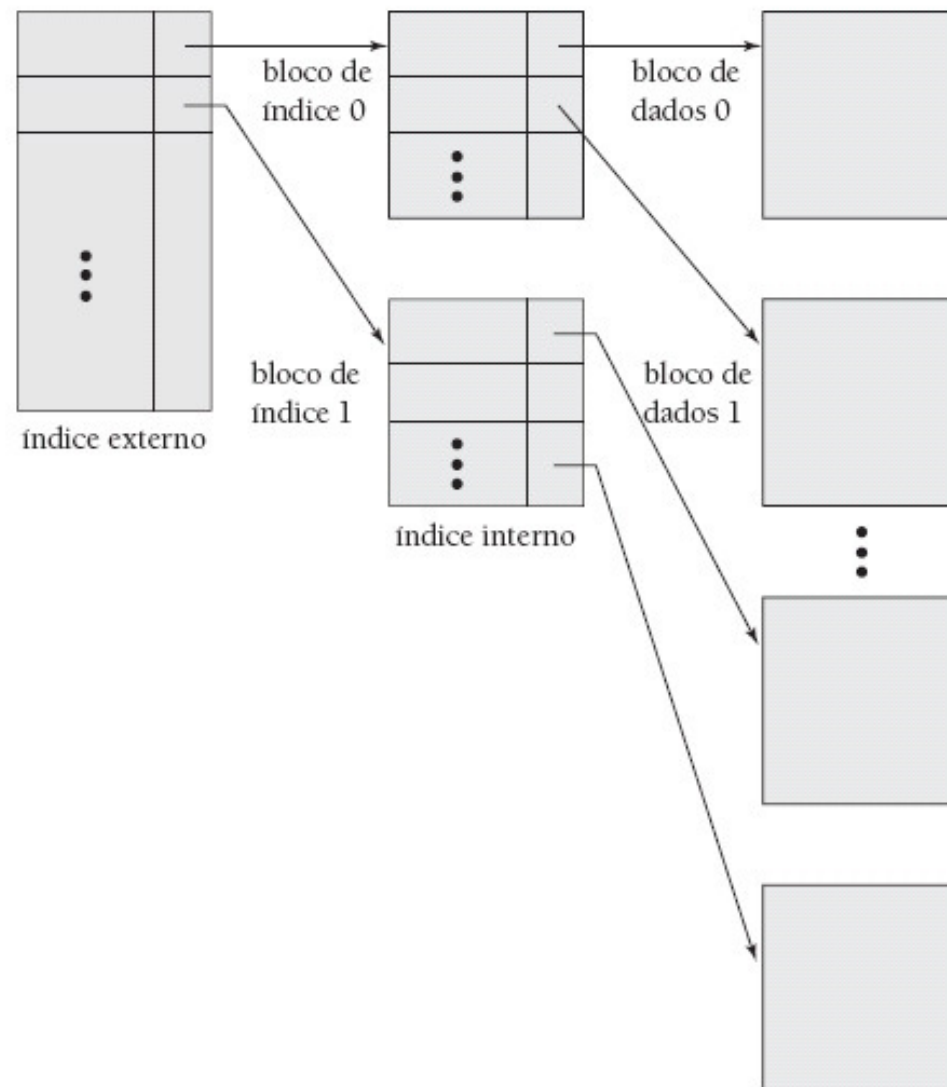
| | | | |
|-------|------------|-----|--|
| A-217 | Brighton | 750 | |
| A-101 | Downtown | 500 | |
| A-110 | Downtown | 600 | |
| A-215 | Mianus | 700 | |
| A-102 | Perryridge | 400 | |
| A-201 | Perryridge | 900 | |
| A-218 | Perryridge | 700 | |
| A-222 | Redwood | 700 | |
| A-305 | Round Hill | 350 | |



Índice Multinível

- Se o índice primário não couber na memória, o acesso se torna dispendioso;
- Para reduzir o número de acessos de disco aos registros de índice, trate o índice primário mantido em disco como um arquivo sequencial e construa um índice esparsos sobre ele.
 - **Índice Externo** – um índice esparsos do índice primário;
 - **Índice Interno** – o arquivo de índice primário;
- Se até mesmo o índice externo for muito grande para caber na memória principal, outro nível de índice pode ser criado, e assim por diante;
- Os índices em todos os níveis precisam ser atualizados na inserção ou exclusão no arquivo.

Índice Multinível



Exclusão – Atualização de Índice

- Se o registro excluído for o único registro no arquivo com seu valor de chave de busca específico, a chave de busca também é excluída do índice;
- Exclusão de índice de único nível:
 - Índices densos - a exclusão da chave de busca é semelhante à exclusão de registro do arquivo;
 - Índices esparsos - se houver uma entrada para a chave de busca no índice, ela é excluída substituindo a entrada no índice pelo próximo valor de chave de busca no arquivo (em ordem de chave de busca). Se o próximo valor de chave de busca já tiver uma entrada de índice, a entrada é excluída em vez de ser substituída.

Inclusão – Atualização de Índice

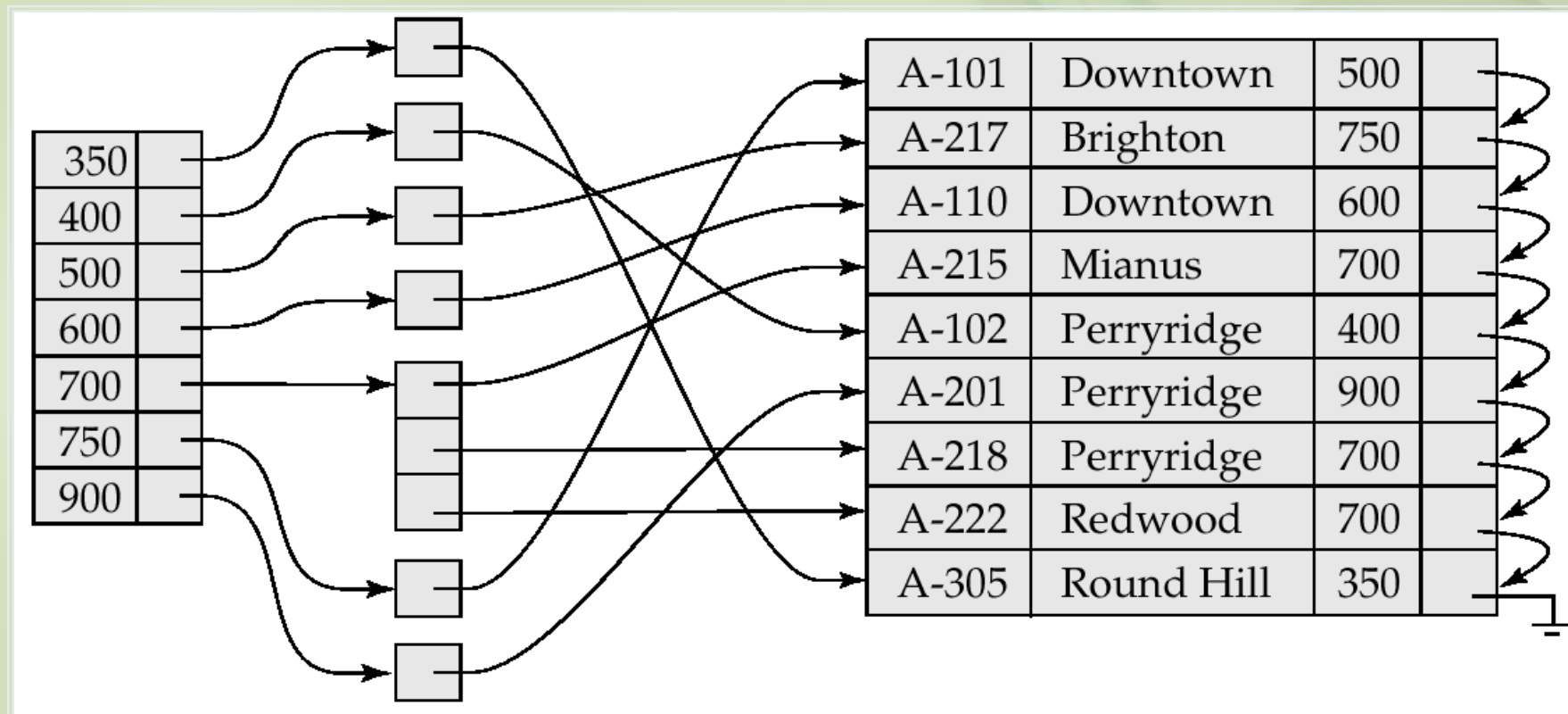
- Inserção de índice de único nível:
 - Realizar uma pesquisa usando o valor de chave de busca que aparece no registro a ser inserido;
 - Índices densos - se o valor da chave de busca não aparecer no índice, insira-o;
 - Índices esparsos - se o índice armazena uma entrada para cada bloco do arquivo, nenhuma mudança precisa ser feita no índice, a menos que um novo bloco seja criado. Nesse caso, o primeiro valor de chave de busca que aparece no novo bloco é inserido no índice;
- Algoritmos de inserção multinível (além de exclusão) são simples extensões dos algoritmos de único nível.

Índices Secundários

- Frequentemente, alguém deseja encontrar todos os registros cujos valores em um certo campo (que não seja a chave de busca do índice primário) satisfazem alguma condição.
 - Exemplo 1: No banco de dados *conta* armazenado sequencialmente por número de conta, podemos querer encontrar todas as contas em determinada agência;
 - Exemplo 2: como antes, mas onde queremos encontrar todas as contas com um saldo especificado ou intervalo de saldos;
- Podemos ter um **índice secundário** com um registro de índice para cada valor de chave de busca; o registro de índice aponta para um arquivo que contém ponteiros para todos os registros reais com esse valor de chave de busca específico.

Índices Secundários

- Exemplo: índice secundário sobre o campo saldo de conta



Índices Primários e Secundários

- Índices secundários precisam ser densos;
- Índices oferecem benefícios substanciais quando procuram registros;
- Quando um arquivo é modificado, cada índice no arquivo precisa ser atualizado. A atualização de índices impõe sobrecarga na modificação do banco de dados;
- A varredura sequencial usando índice primário é eficiente, mas uma varredura sequencial usando um índice secundário é dispendiosa.
 - Cada acesso a registro pode apanhar um novo bloco do disco.

Arquivos de Índice de Árvore B+

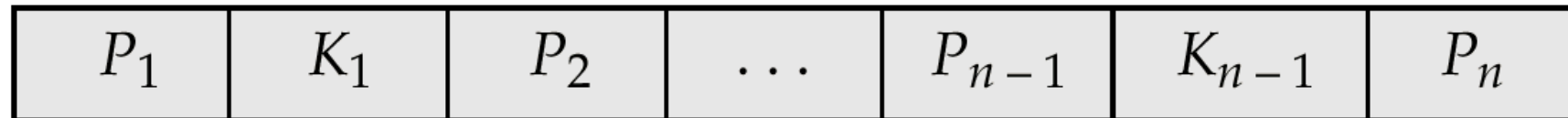
- Desvantagem dos arquivos sequenciais indexados: o desempenho diminui quando o arquivo aumenta, pois muitos blocos de estouro são criados. É preciso reorganizar periodicamente o arquivo inteiro;
- Vantagem dos arquivos de índice B⁺: reorganiza-se automaticamente com pequenas mudanças locais, em face a inserções e exclusões. A reorganização do arquivo inteiro não é necessária para manter o desempenho;
- Desvantagem das árvores B⁺: trabalho extra de inserção e exclusão, sobrecarga de espaço;
- Vantagens das árvores B⁺ superiores às desvantagens, e por isso são muito usadas.

Arquivos de Índice de Árvore B+

- Um índice de árvore B+ é um índice de multinível, mas tem uma estrutura que difere daquela do arquivo sequencial indexado multinível;
- Uma árvore B+ é uma árvore com raiz satisfazendo as seguintes propriedades:
 - Todos os caminhos da raiz até a folha são do mesmo tamanho;
 - Cada nó que não é uma raiz ou uma folha tem entre $\lceil n/2 \rceil$ e n filhos;
 - Um nó de folha tem entre $\lceil (n-1)/2 \rceil$ e $n-1$ valores;
- Casos especiais:
 - Se a raiz não for uma folha, ela tem pelo menos 2 filhos;
 - Se a raiz for uma folha (ou seja, não houver outros nós na árvore), ela pode ter entre 0 e $(n-1)$ valores.

Arquivos de Índice de Árvore B+

- Estrutura típica:

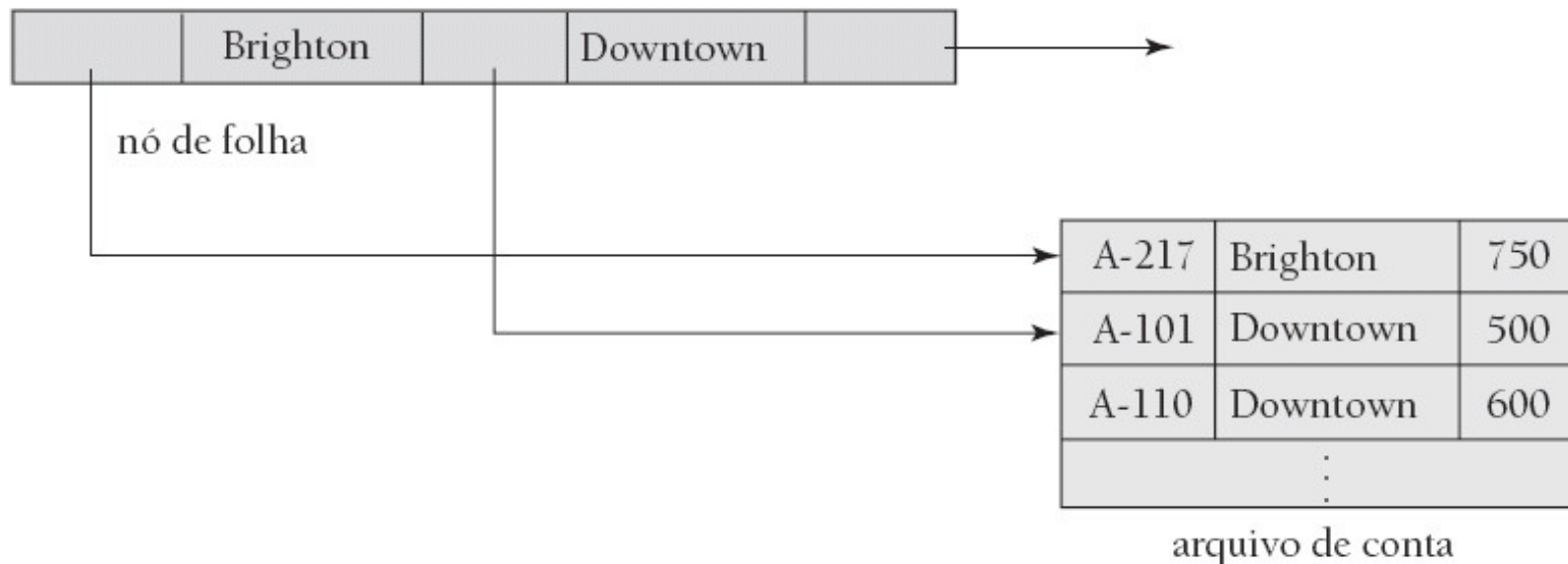


- K_i são os valores de chave de busca;
- P_i são ponteiros para os nós-filhos (para nós não de folha) ou ponteiros para registros ou baldes de registros (para nós de folha);
- As chaves de busca em um nó são ordenadas:

$$K_1 < K_2 < K_3 < \dots < K_{n-1}$$

Arquivos de Índice de Árvore B+

- Para $i = 1, 2, \dots, n-1$, o ponteiro P_i ou aponta para um registro de arquivo com valor de chave de busca K_i , ou para um balde de ponteiros para registros de arquivo, cada registro tendo valor de chave de busca K_i . Só é preciso a estrutura de balde se a chave de busca não formar uma chave primária;
- Se L_i, L_j forem nós de folha e $i < j$, os valores de chave de busca de L_i são menores que os valores de chave de busca de L_j ;
- P_n aponta para o próximo nó de folha na ordem da chave de busca.



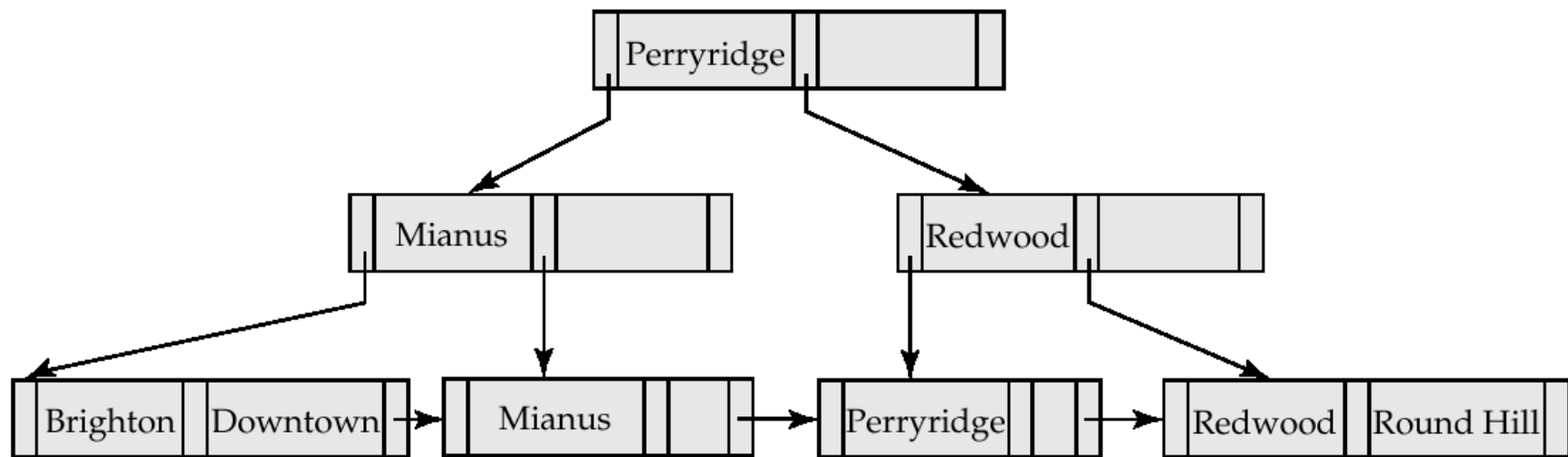
Arquivos de Índice de Árvore B+

- Os nós não de folha formam um índice esparsos multinível sobre os nós de folha. Para um nó não de folha com m ponteiros:
 - Todas as chaves de busca na sub-árvore à qual P_1 aponta são menores que K_1 ;
 - Para $2 \leq i \leq n-1$, todas as chaves de busca na sub-árvore à qual P_i aponta têm valores maiores ou iguais a K_{i-1} e menores que K_{m-1} .

| | | | | | | |
|-------|-------|-------|---------|-----------|-----------|-------|
| P_1 | K_1 | P_2 | \dots | P_{n-1} | K_{n-1} | P_n |
|-------|-------|-------|---------|-----------|-----------|-------|

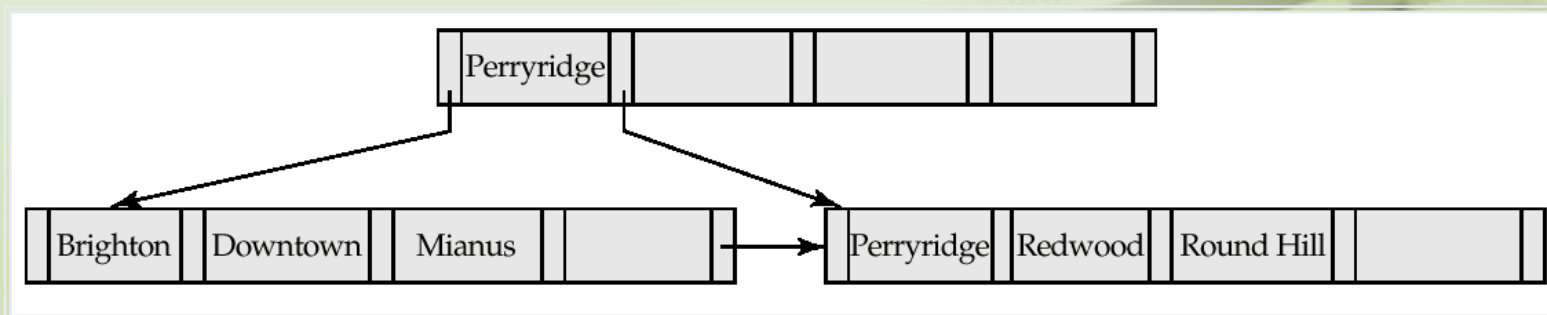
Arquivos de Índice de Árvore B+

- Exemplo 1 - Árvore B+ para arquivo *conta* ($n = 3$).



Arquivos de Índice de Árvore B+

- Exemplo 2 -Árvore B+ para arquivo *conta* ($n = 5$).
 - Nós de folha precisam ter entre 2 e 4 valores ($\lceil (n-1)/2 \rceil$ e $n-1$, com $n = 5$);
 - Nós não de folha diferentes da raiz precisam ter entre 3 e 5 filhos ($\lceil n/2 \rceil$ e n com $n=5$);
 - Raiz precisa ter pelo menos 2 filhos.



Arquivos de Índice de Árvore B+

- Observações:
 - Como as conexões entre nós são feitas por ponteiros, blocos “logicamente” próximos não precisam estar “fisicamente” próximos;
 - Os níveis não de folha da árvore B⁺ formam uma hierarquia de índices esparsos;
 - A árvore B⁺ contém uma quantidade relativamente pequena de níveis (logarítmica no tamanho do arquivo principal), de modo que as buscas podem ser conduzidas de modo eficiente;
 - Inserções e exclusões no arquivo principal podem ser tratadas de modo eficiente, pois o índice pode ser reestruturado em tempo logarítmico.

Consulta em Árvore B+

- Encontre todos os registros com um valor de chave de busca k .
- Comece com o nó raiz.
 - Examine o nó em busca do menor valor de chave de busca $> k$;
 - Se esse valor existir, considere que é K_j . Depois siga P_i até o nó filho;
 - Caso contrário, $k \geq K_{m-1}$, onde existem m ponteiros no nó. Depois, siga P_m até o nó filho;
- Se o nó alcançado seguindo-se o ponteiro acima não for um nó de folha, repita o procedimento sobre o nó e siga o ponteiro correspondente;
- Por fim, alcance um nó de folha. Se, para algum i , a chave $K_i = k$, siga o ponteiro P_i até o registro ou balde desejado. Se não, não há um registro com valor de chave de busca k .

Consulta em Árvore B+

- No processamento de uma consulta, um caminho é atravessado na árvore da raiz até algum nó de folha;
- Se houver K valores de chave de busca no arquivo, o caminho não será maior do que $\lceil \log_{\lceil n/2 \rceil}(K) \rceil$;
- Um nó geralmente tem o mesmo tamanho de um bloco de disco, normalmente 4 kilobytes, e n normalmente fica em torno de 100 (40 bytes por entrada de índice);
- Com 1 milhão de valores de chave de busca e $n = 100$, no máximo $\log_{50}(1.000.000) = 4$ nós são acessados em uma pesquisa;
- Compare isso com uma árvore binária balanceada com 1 milhão de valores de chave de busca - cerca de 20 nós são acessados em uma pesquisa.
- A diferença acima é significativa, pois o acesso a cada nó pode precisar de uma E/S de disco, custando em torno de 20 milissegundos!

Inserção em Árvore B+

- Encontre o nó de folha em que o valor da chave de busca deveria aparecer;
- Se o valor da chave de busca já estiver lá no nó de folha, o registro é acrescentado ao arquivo e, se for preciso, um ponteiro é inserido no balde;
- Se o valor da chave de busca não estiver lá, então acrescente o registro ao arquivo principal e crie um balde, se for preciso. Depois:
 - Se houver espaço no nó de folha, insira o par (valor de chave, ponteiro) no nó de folha;
 - Caso contrário, divida o nó (junto com a nova entrada (valor de chave, ponteiro)), conforme a seguir.

Inserção em Árvore B+

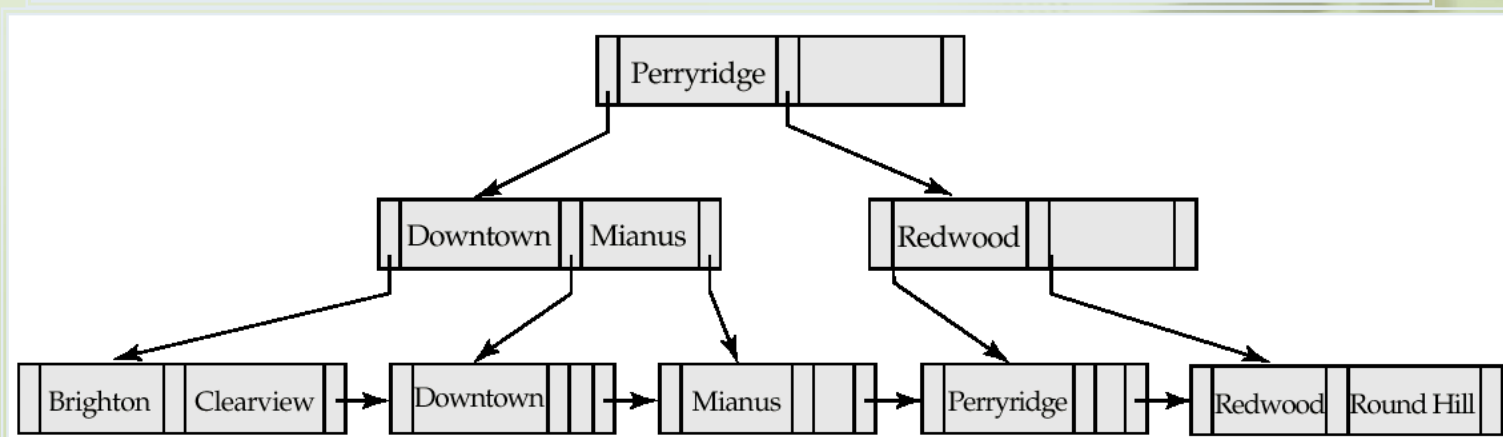
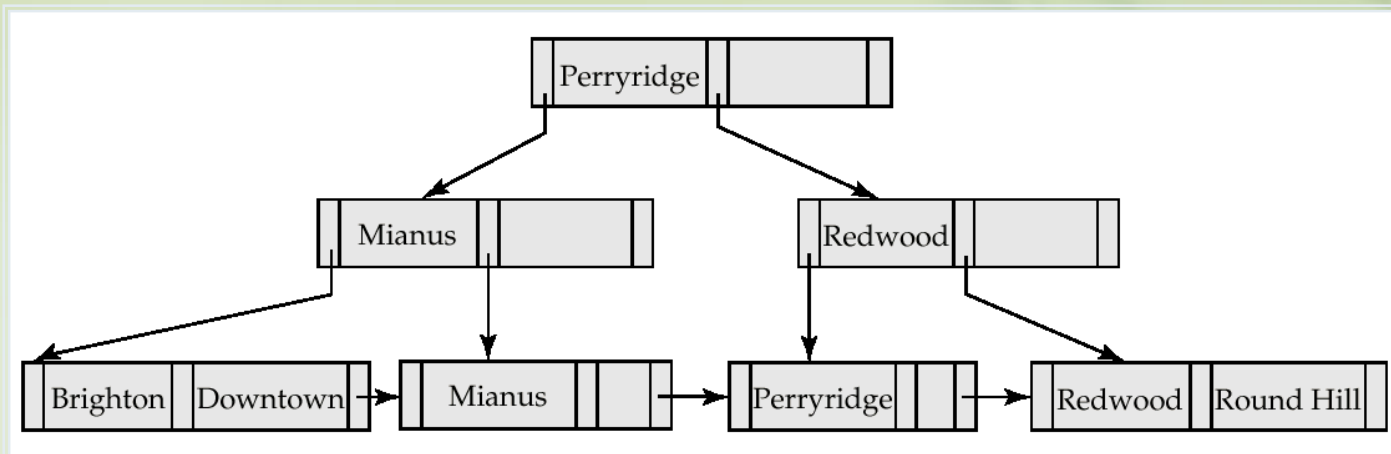
- Dividindo um nó:
 - Apanhe os n pares (valor de chave, ponteiro) (incluindo aquele sendo inserido) em ordem classificada. Coloque os $\lceil n/2 \rceil$ primeiros no nó original, e o restante em um novo nó;
 - considere que o novo nó seja p , e considere que k é o menor valor de chave em p . Insira (k, p) no pai do nó sendo dividido. Se o pai estiver cheio, divida-o e propague a divisão para cima;
- A divisão de nós prossegue para cima até que um nó que não esteja cheio seja encontrado. No pior caso, o nó raiz pode ser dividido, aumentando a altura da árvore em 1;
- Exemplo:



Resultado da divisão do nó contendo Brighton e Downtown na inserção de Clearview

Inserção em Árvore B+

- Exemplo (cont) - Árvore B+ antes e depois da inserção de "Clearview".

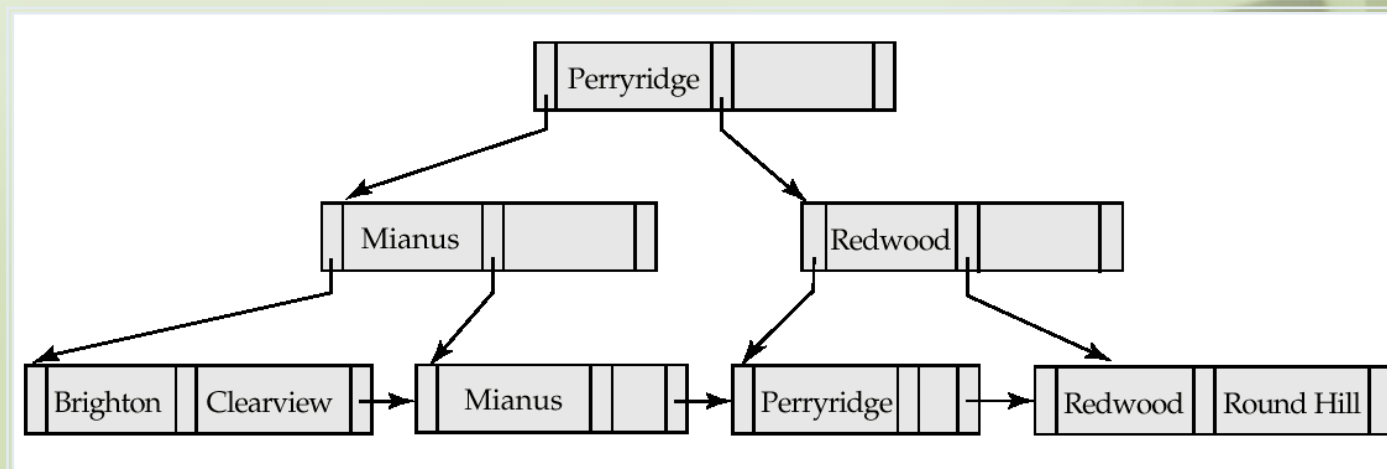
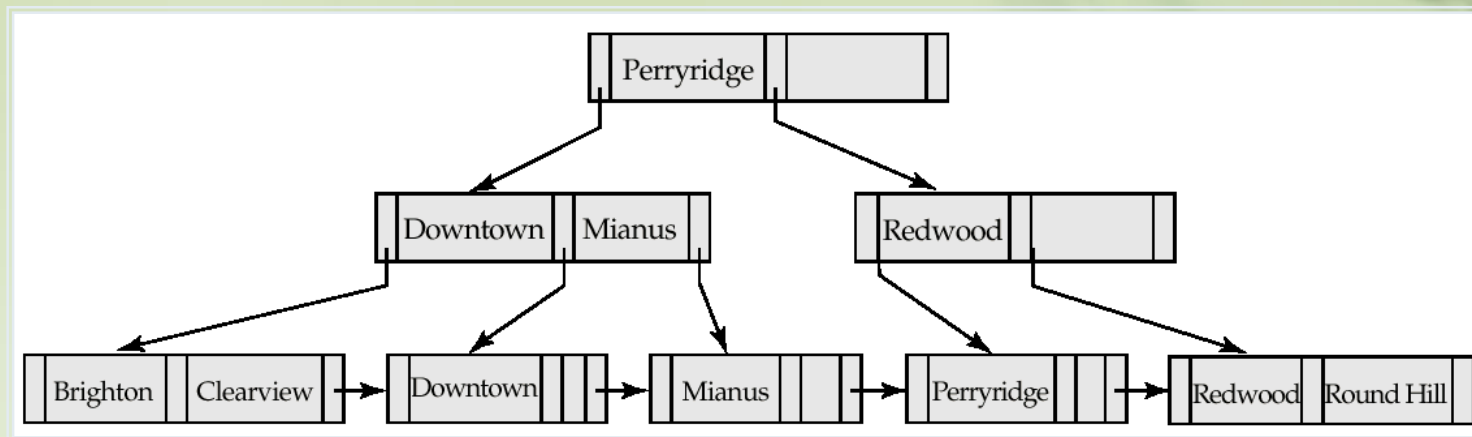


Exclusão em Árvore B+

- Encontre o registro a ser excluído e remova-o do arquivo principal e do balde (se estiver presente);
- Remova o par (valor chave de busca, ponteiro) do nó de folha se não houver balde ou se o balde tiver ficado vazio;
- Se o nó tiver muito poucas entradas devido à remoção, e as entradas no nó e um irmão couberem em um único nó, então:
 - Insira todos os valores de chave de busca dos dois nós em um único nó (aquele à esquerda) e exclua o outro nó;
 - Exclua o par (K_{i-1}, P_i) , onde P_i é o ponteiro para o nó excluído, a partir de seu pai, seguindo recursivamente o procedimento anterior;
- Caso contrário, se o nó tiver muito poucas entradas devido à remoção, e as entradas no nó e um irmão não couberem em um único nó, então
 - Redistribua os ponteiros entre o nó e um irmão, de modo que ambos tenham mais do que o número mínimo de entradas;
 - Atualize o valor de chave de busca correspondente no pai do nó;
- As exclusões de nó podem ser propagadas para cima, até que seja encontrado um nó contendo $\lceil n/2 \rceil$ ou mais ponteiros. Se o nó raiz tiver apenas um ponteiro após a exclusão, ele é excluído e o único filho se torna a raiz.

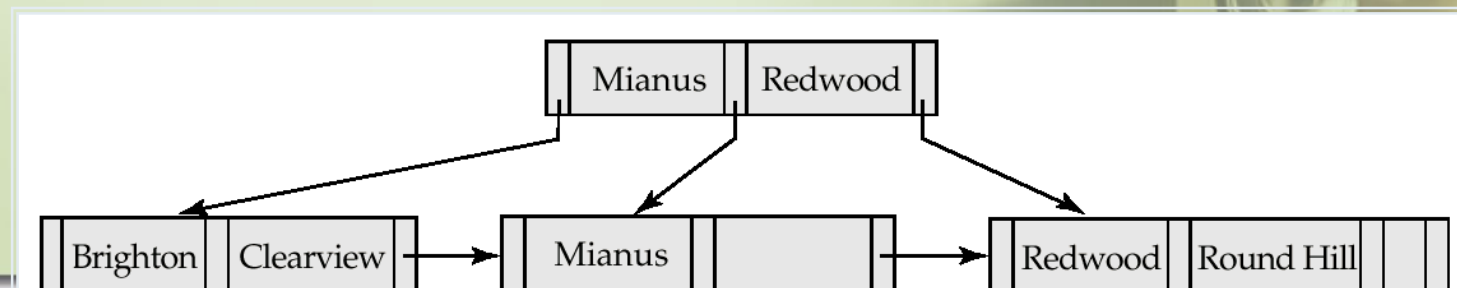
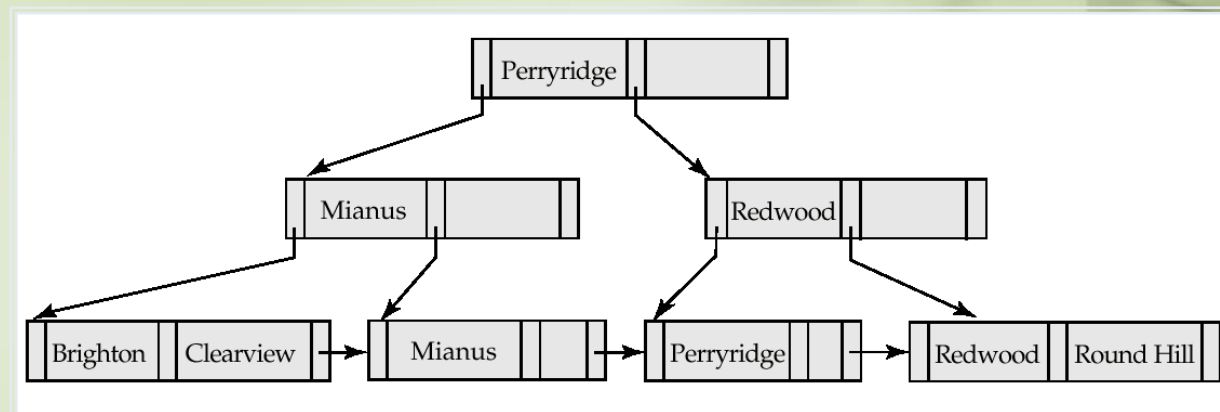
Exclusão em Árvore B+

- Exemplo: A remoção do nó de folha contendo “Downtown” não resultou em seu pai tendo muito poucos ponteiros. Assim, as exclusões em cascata pararam com o pai do nó de folha excluído.



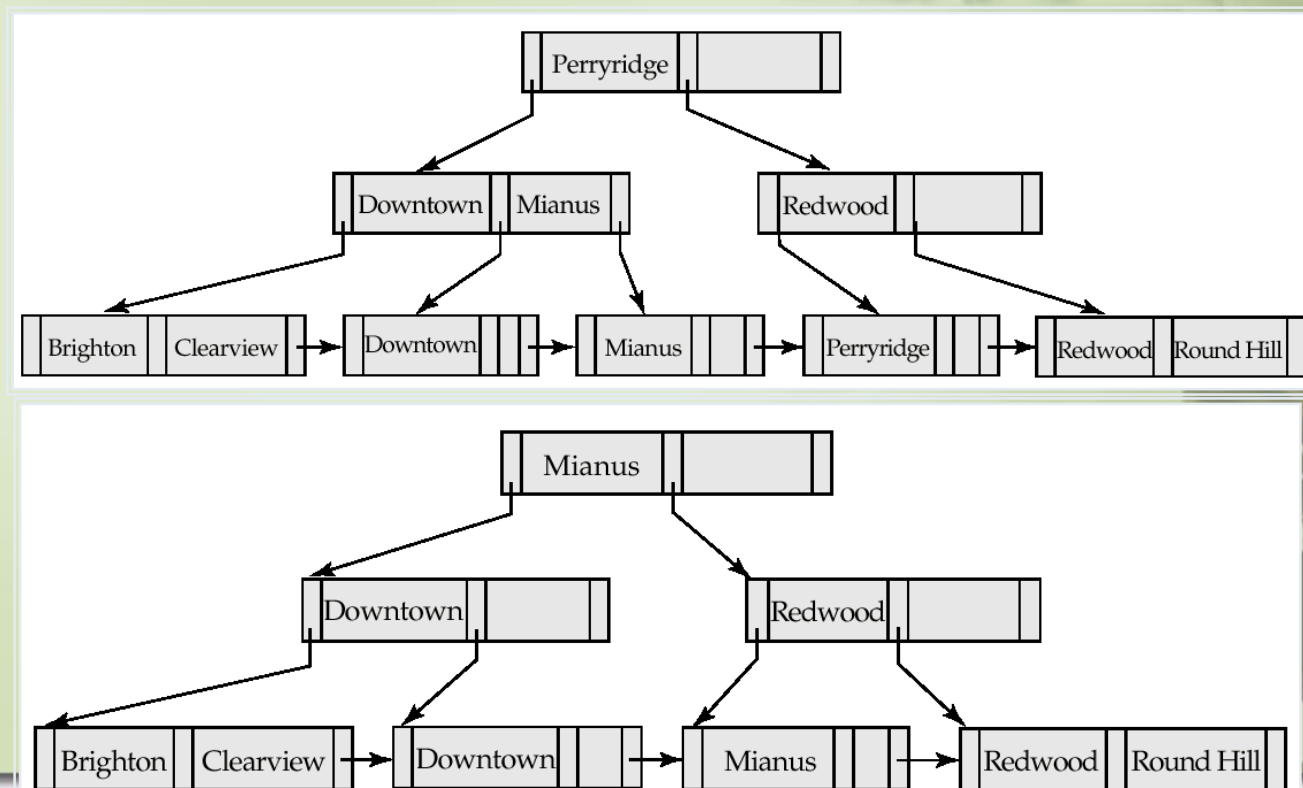
Exclusão em Árvore B+

- Exemplo (cont):
 - Nó com “Perryridge” torna-se menos que cheio (na verdade, vazio, nesse caso especial) e mesclado com seu irmão;
 - Como resultado, o pai do nó “Perryridge” se torna menos que cheio, e foi mesclado com seu irmão (e uma entrada foi excluída do seu pai);
 - O nó raiz, então, tinha apenas um filho, e foi excluído e seu filho se tornou o novo nó raiz.



Exclusão em Árvore B+

- Exemplo (cont): Antes e depois da exclusão de “Perryridge” do exemplo anterior.
- Pai da folha contendo Perryridge tornou-se menos que cheio, e pediu emprestado um ponteiro do seu irmão da esquerda;
- Valor de chave de busca no pai do pai muda como resultado.

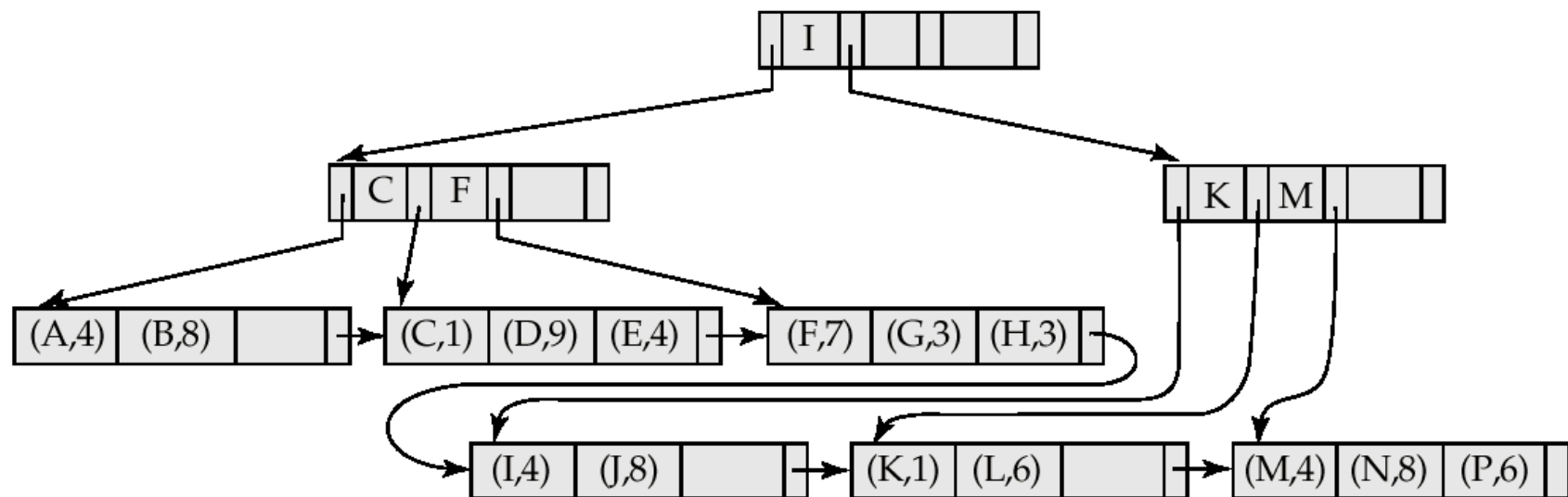


Organização de Arquivo em Árvore B+

- O problema da degradação do arquivo de índice é resolvido com o uso de índices de árvore B+. O problema da degradação do arquivo de dados é resolvido com o uso da organização de arquivo de árvore B+;
- Os nós de folha em uma organização de arquivo de árvore B+ armazenam registros, ao invés de ponteiros;
- Como os registros são maiores que os ponteiros, o número máximo de registros que podem ser armazenados em um nó de folha é menor que o número de ponteiros em um nó não de folha;
- Os nós de folha ainda precisam estar meio cheios;
- A inserção e a exclusão são tratadas da mesma maneira que a inserção e a exclusão de entradas em um índice de árvore B+.

Organização de Arquivo em Árvore B+

- Exemplo de organização de arquivo de árvore B+.
- Boa utilização de espaço importante, pois os registros usam mais espaço que os ponteiros;
- Para melhorar a utilização de espaço, envolva mais nós irmãos na redistribuição durante divisões e mesclagens.
 - Envolver 2 irmãos na redistribuição (para evitar a divisão/mesclagem onde possível) resulta em cada um tendo pelo menos $\lfloor 2n/3 \rfloor$ entradas.



Arquivos de Índice de Árvore B

- Semelhantes à árvore B+, mas a árvore binária só permite que valores de chave de busca apareçam uma vez; elimina o armazenamento redundante de chaves de busca;
- Chaves de busca em nós não de folha não aparecem em outro lugar nas árvores binárias; um campo de ponteiro adicional para cada chave de busca em um nó não de folha precisa ser incluído;
- Nó de folha de árvore binária generalizado.



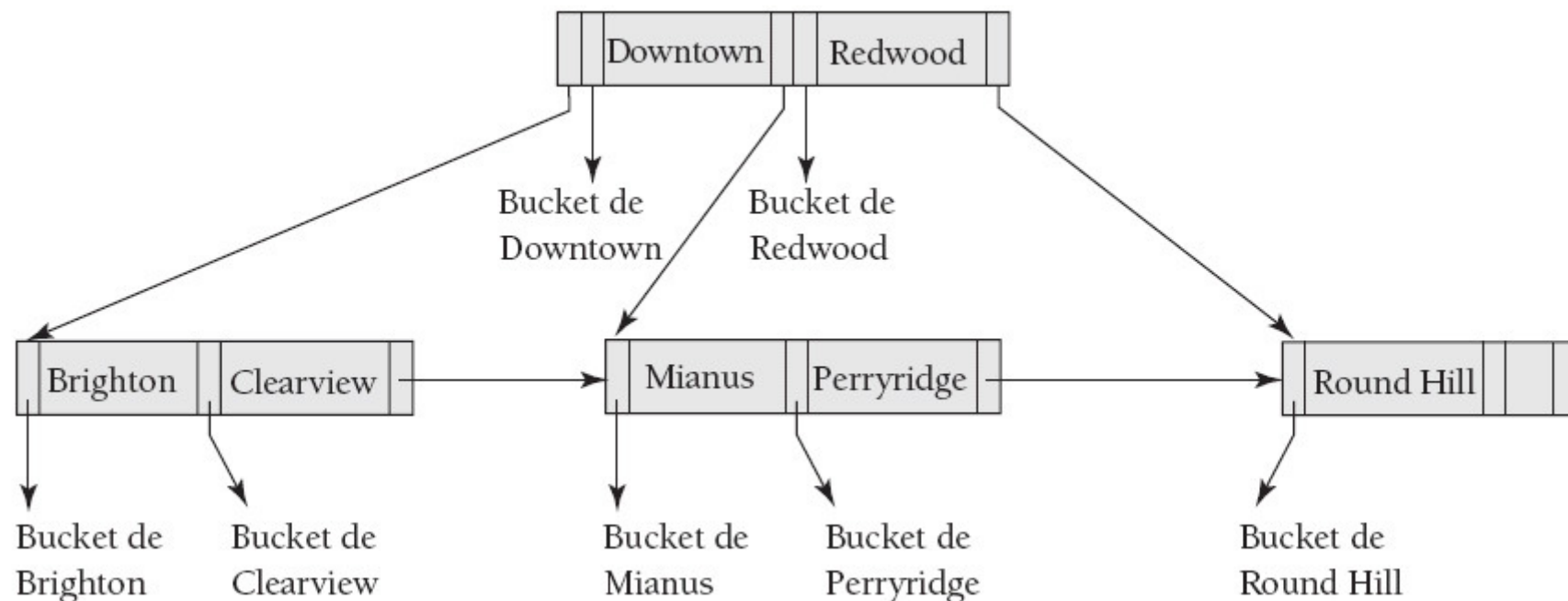
(a)



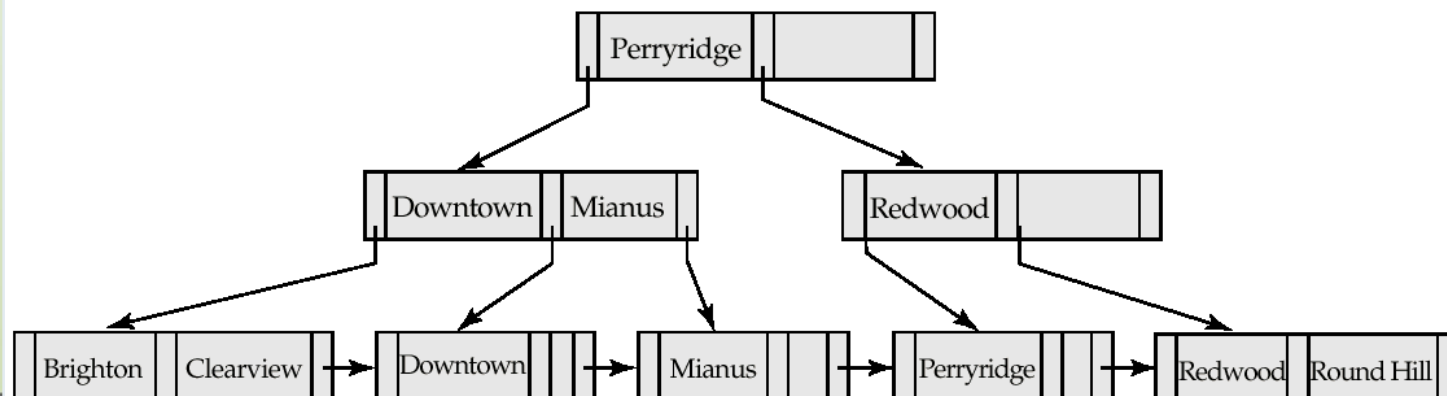
(b)

- Nó não de folha – ponteiros B_i são o balde dos ponteiros de registros de arquivo.

Arquivos de Índice de Árvore B - Exemplo



**Árvore binária
(acima) e árvore
B+ (ao lado)
sobre os
mesmos dados**



Arquivos de Índice de Árvore B

- Vantagens dos índices de árvore binária:
 - Podem usar menos nós de árvore que uma árvore B⁺ correspondente;
 - Às vezes é possível encontrar o valor da chave de busca antes de alcançar o nó de folha;
- Desvantagens dos índices de árvore binária:
 - Apenas uma pequena fração de todos os valores de chave de busca é encontrada cedo;
 - Os nós não de folha são maiores, de modo que o fan-out é reduzido. Assim, as árvores binárias normalmente possuem maior profundidade que a árvore B⁺ correspondente;
 - A Inserção e a exclusão mais complicadas do que nas árvores B⁺;
 - A implementação é mais difícil do que nas árvores B⁺;
- Normalmente, as vantagens das árvores binárias não superam as desvantagens, em comparação com as árvores B⁺.

SQL – Definição de Índice

- Criar um índice:

```
create index <nome-índice> on <nome-tabela>
                        (<lista-atributos>)
```

- **Por exemplo:** `create index índice-b on agência (nome-agência)`
- Usar `create unique index` para especificar indiretamente e impor a condição em que a chave de busca é uma chave candidata.
 - Não é realmente necessário se a restrição de integridade `unique` da SQL for admitida;
- Para remover um índice:

```
drop index <nome-índice>
```

Acesso de Chave Múltipla

- É interessante usar índices múltiplos para certos tipos de consultas;
- Exemplo:

```
SELECT número_conta  
FROM conta  
WHERE nome_agência = "Perryridge" AND saldo = 1000
```

- Estratégias possíveis para processar consulta usando índices sobre atributos isolados:
 1. Use índice sobre *nome_agência* para encontrar contas com saldos de \$1000. teste *nome_agência* = "Perryridge";
 2. Use índice sobre *saldo* para encontrar contas com saldos de \$1000. teste *nome_agência* = "Perryridge".
 3. Use índice *nome_agência* para encontrar ponteiros para todos os registros pertencentes à agência Perryridge. De modo semelhante, use índice sobre *saldo*. Apanhe a interseção dos dois conjuntos de ponteiros obtidos.

Índices sobre Atributos Múltiplos

- Suponha que temos um índice sobre a chave combinada (*nome_agência*, *saldo*);
- Com a cláusula where
where *nome_agência* = "Perryridge" and *saldo* = 1000
o índice sobre a chave de busca combinada apanhará apenas registros que satisfazem as duas condições.
O uso de índices separados é menos eficiente - podemos apanhar muitos registros (ou ponteiros) que satisfazem apenas uma das condições.
Também pode tratar de modo eficiente
where *nome_agência* = "Perryridge" and *saldo* < 1000
- Mas não pode tratar de modo eficiente
where *nome_agência* < "Perryridge" and *saldo* = 1000
Pode apanhar muitos registros que satisfazem a primeira, mas não a segunda condição.

Para estudos

- Lista de Exercícios (disponibilizada no site);
- Leitura:
SILBERSCHATZ, A. et.al. Sistema de banco de dados. 5.ed. Rio de Janeiro: Elsevier, 2006. Capítulo 12 – Indexação e Hashing.